BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

Objek penelitian adalah hal awal yang harus ditentukan dalam kegiatan penelitian sehingga penelitian dapat dilakukan secara efektif dan efisien sesuai dengan tujuan penelitian. Pemilihan dan penentuan objek penelitian yang tepat diharapkan dapat menunjang kegiatan selama penelitian, sehingga hal-hal yang diperlukan dalam penelitian akan mudah dicapai.

Pengertian objek penelitian secara umum merupakan permasalahan yang dijadikan topic penulisan dalam rangka menyusun suatu laporan. Penelitian ini dilakukan untuk memperoleh data-data yang berkaitan dengan objek penelitian yang telah penulis lakukan.

3.1.1 Sejarah TK Negeri Pembina

Taman Kanak-kanak Negeri Pembina kabupaten Bandung merupakan TK Percontohan bagi TK-TK yang lain, baik dalam sarana prasarana maupun dalam program dan proses pembelajaran. Pada awalnya dibina langsung oleh Dinas Pendidikan dan Kebudayaan Provinsi Jawa Barat bidang Pendidikan Dasar. Namun dengan adanya kebijakan otonomi daerah maka pembinaan dan pengelolaan selanjutnya diserahkan kepada Dinas Pendidikan dan Kebudayaan kabupaten Bandung.Pendidikan di TK Negeri Pembina di titik beratkan pada pembentukan watak dan pribadi mandiri agar siap belajar ke jenjang pendidikan selanjutnya dan lebih siap menghadapi dunia baru.

3.1.2 Visi Misi

Adapun Visi Misi dari TK Negeri Pembina adalah :

Visi

" Membentuk generasi yang beriman dan bertaqwa, berbudi luhur dan berakhlak mulia,cerdas, terampil dalam menghadapi masa depan."

Misi

1. Mewujudkan keimanan sdan ketaqwaan anak melalui pembiasaan dalam kegiatan agama
2. Mengembangkan potensi anak didik yang positif untuk kemandirian dalam kehidupan sehari hari.
3. Memberi pengalaman bagi anak-anak untuk mengembangkan pengetahuan, keterampilan dan bertanggung jawab.

3.1.3 Tujuan TK Negeri Pembina

Secara umum Taman Kanak-Kanak Negeri Pembina Kabupaten Bandung mempunyai tujuan untuk mengembangkan kemampuan dan membentuk watak serta peradaban bangsa yang bermanfaat dalam rangka mencerdaskan kehidupan bangsa, untuk berkembangnya potensi peserta didik agar menjadi manusia yang beriman kepada Tuhan Yang Maha Esa, berakhlak mulia, sehat, berilmu, cakap, kreatif, mandiri dan menjadi warga negara yang demokratis serta bertanggung jawab. Sedangkan secara khusus yang menjad tujuan sekolah, adalah :

1. Agar menjadi anak yang beriman dan bertaqwa kepada Tuhan Y.M.E serta berakhlak mulia.

2. Menjadikan output TKN Pembina menjadi mandiri dan bertanggung jawab kepada tugas.
3. Anak siap dan mau menerima pendidikan pada jenjang selanjutnya.

Profil Sekolah

NamaTK	:	TK Negeri Pembina
Status Tk	:	Negeri
Alamat	:	Jl. Cibiru Raya KM.17
Kode Pos	:	
Desa	:	Cibiru Wetan
Kecamatan	:	Cileunyi
Kabupaten	:	Bandung
No. Tlp. Yang dapat dihubungi	:	022-87822991
A. Tahun Pendirian	:	1997
B. Tahun Beroperasi	:	1998

Jumlah Anak Didik Dalam 3 Tahun Terakhir

Kelompok	**Jumlah Anak Didik**			**Keterangan**
	2009/2010	**2010/2011**	**2011/2012**	
A	40	41	41	
B	85	109	109	
Jumlah	125	150	150	

Tabel 3.1 Jumlah Anak Didik

Tenaga Kependidikan

Nama Guru	Status	Pendidikan	Pangkat/ Golongan	Umur	Masa Kerja
1. Eli Carwati, S, Pd	PNS	S-1	III/d	44	15 Tahun
2. Titin Hayatin	PNS	D-2	III/d	47	21 Tahun
3. Enung Wartini	PNS	D-2	II/c	37	8 Tahun
4. Ida, A.Ma	PNS	D-2	II/b	36	2 Tahun
5. Fitrya Yulianty, A.Ma	PNS	D-2	II/b	26	2 Tahun
6. Yulia Triani	PNS	SPGTK	II/a	35	2 Tahun
7. Susy Indrawati, A.Ma	PNS	D-2	II/b	36	2 Tahun
8. Herlina Kostaman, A. Ma	PNS	SPGTK	II/a	40	2 Tahun
9. Yetti Kurniawati, S. Pd	PNS	SPGTK	II/a	41	1 Tahun

Tabel 3.2 Tenaga Kependidikan

Potensi Yang Pernah Diraih

No.	Prestasi				Keterangan
	Lembaga TK	Kepala	Guru	Anak Didik	
1.		Kinerja			Tk. Provinsi
2.			Guru Prestasi		Tk. Kabupaten
3.				Mewarnai	Tk. Kabupaten

Tabel 3.3 Potensi Yang Pernah Diraih

3.2 Metodologi Yang Digunakan

Metode penelitian adalah suatu teknik atau cara mengumpulkan, memperoleh data, baik berupa data primer maupun sekunder yang digunakan untuk keperluan menyusun suatu karya ilmiah dan kemudian menganalisa faktor-faktor yang berhubungan dengan pokok-pokok permasalahan sehingga terdapat suatu kebenaran data-data yang akan diperoleh. Adapun pengertian metode penelitian menurut Sugiyono (2010 : 2), diantaranya adalah sebagai berikut : *" Metode penelitian pada dasarnya merupakan cara ilmiah untuk mendapatkan data dengan tujuan dan kegunaan tertentu"*.

3.2.1 Desain Penelitian

Dalam melakukan suatu penelitian sangat perlu dilakukan perencanaan dan perancangan penelitian, agar penelitian yang dilakukan dapat berjalan dengan baik dan sistematis.

Desain penelitian menurut Moh. Nazir (2003 : 84) memaparkan bahwa "Desain penelitian adalah semua proses yang diperlukan dalam perencanaan dan pelaksanaan penelitian".

Dari definisi di atas maka dapat dikatakan bahwa desain penelitian merupakan semua proses penelitian yang dilakukan oleh penulis dalam melaksanakan penelitian mulai dari perencanaan sampai dengan pelaksanaan penelitian yang dilakukan pada waktu tertentu.

3.2.2 Jenis dan Metode Pengumpulan Data

Metode pengumpulan data adalah penelitian lapangan (*field Research*), dilakukan dengan cara mengadakan peninjauan langsung pada instansi yang menjadi objek untuk mendapatkan data primer (data yang diperoleh dari TK Negeri Pembina) dan data sekunder.

3.2.2.1 Sumber Data Primer (Wawancara, Observasi, Kuesioner)

Data primer merupakan data yang diambil secara langsung, data ini diperoleh dari kegiatan observasi yaitu pengamatan langsung pada objek penelitian dan mengadakan wawancara dengan pihak terlibat.

a. Observasi

Dilakukan dalam bentuk pengamatan secara langsung terhadap pelaksanaan pembuatan sebuah aplikasi tutorial itu sendiri.

b. Wawancara

Pengumpulan data yang dilakukan dengan cara tanya jawab dengan unsur individu (guru) yang terlibat guna mendapaktan data yang akurat.

3.2.2.2 Sumber Data Sekunder (dokumentasi)

Dengan memperoleh dokumen – dokumen yang bersangkutan dengan obyek yang diteliti, yang dimaksudkan sebagai bukti bahwa penelitian benar – benar dilakukan pada perusahaan yang bersangkutan.

3.2.3 Metoda Pendekatan Sistem dan Pengembangan Sistem

3.2.3.1 Metode Pendekatan Sistem

Dalam melakukan penelitian ini, penulis menggunakan pendekatan sistem prototipe. Dimana di dalamnya terdapat identifikasi komponen – komponen yang akan dirancang secra rinci yang bertujuan untuk memberikan gambaran kepada pengguna atau user mengenai sistem.

Adapun tahapan perancangan adalah agar pada akhir pengembangan perangkat lunak dapat memenuhi kebutuhan *user*, efektif, efisien, mudah digunakan,

Tahapan perancangan:

1. Pengumpulan kebutuhan
2. Membangun prototype
3. Evaluasi prototype
4. Pengkodean sistem
5. Menguji sistem

6. Evaluasi sistem

3.2.3.1.1 OOP (Object Oriented Programing)

OOP adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrograman prosedural sudah tidak layak lagi (Efendi Hariyadi) .

OOP adalah suatu model pemrograman yang berorientasi pada obyek. Maksudnya adalah, bahwa setiap bagian dari apa yang anda program -visual maupun non visual- adalah sebuah obyek. Obyek, dalam konteks ini adalah "sesuatu", bisa dianalogikan dengan benda, yang padanya melekat attribut-attribut. Attribut tersebut bisa dikelompokkan ke dalam 2 jenis yaitu kelompok "property" dan kelompok "method" atau "event"(Y. Wibisono).

OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep dari OOP sendiri adalah, semua pemecahan masalah dibagi ke dalam objek. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti inheritance atau penurunan, encapsulation atau pembungkusan.

Empat konsep yang dimiliki oleh pemograman yang berorientasi object adalah :

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

3.2.3.1.1.1 Encapsulation

Encapsulation adalah penyatuan antara proses-proses yang terjadi dan sejumlah data yang dikelola kedalam suatu bentuk kelompok tertentu.

3.2.3.1.1.2 Abstraction

Abstraction memiliki pengertian bahwa dalam perangkat lunak seorang programmer akan menggunakan atau menciptakan sejumlah objek dimana objek tersebut ada dan berprilaku pada sistem.

3.2.3.1.1.3 Inheritance

Inheritance atau pewarisan adalah merupakan konsep dimana suat objek memiliki sejumlah perilaku yang dimiliki oleh objek lain dalam suatu hirarki tertentu.

3.2.3.1.1.4 Polymorphism

Polymorphism adalah konsep dimana beberapa objek dapat memiliki metode yang sama dengan aksi yang berbeda.

3.2.3.1.2 UML (Unified Modelling Language)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah system(Sri Dharwijayanti, Romi Satria Wahono).

The Unified Modeling Language (UML) adalah bahasa standart untuk melakukan spesifikasi, visualisasi, konstruksi, dan dokumentasi dari komponen-komponen perangkat lunak, dan digunakan untuk pemodelan bisnis(Tessy Badriyah, SKom. MT.).

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

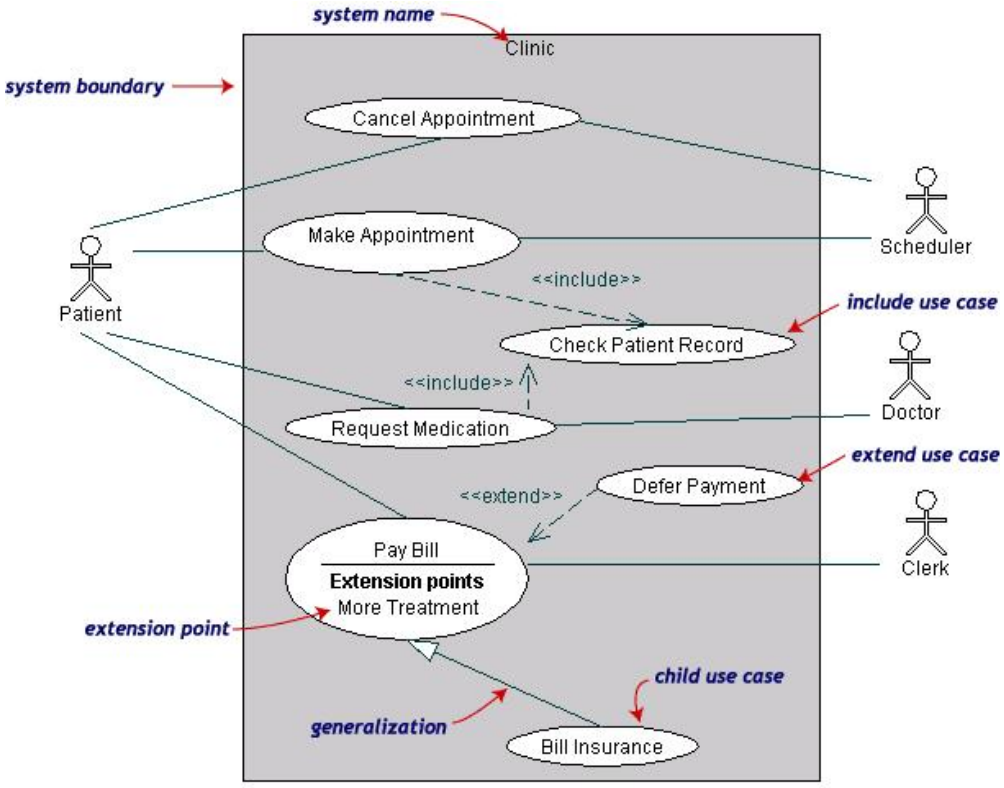
UML mendefinisikan diagram-diagram sebagai berikut:

1. use case diagram
2. class diagram
3. statechart diagram

4. activity diagram
5. sequence diagram
6. collaboration diagram
7. component diagram
8. deployment diagram

3.2.3.1.2.1 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.



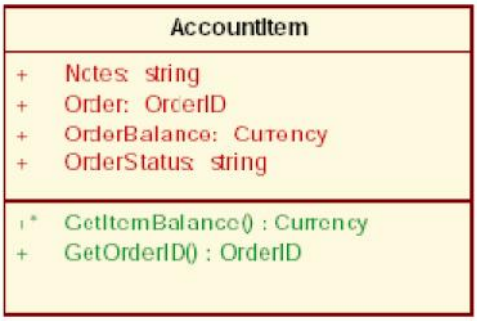
Gambar 3.1 Contoh Use Case Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.3.1.2.2 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda



Gambar 3.2 Contoh Class Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

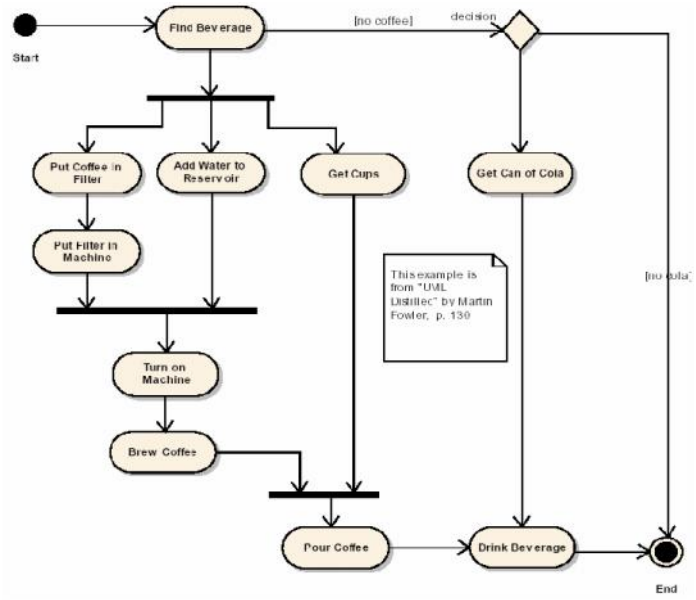
3.2.3.1.2.3 Statechhart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

3.2.3.1.2.4 Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*).

Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.



Gambar 3.3 Contoh Activiy Diagram
(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.3.1.2.5 Sequence Diagram

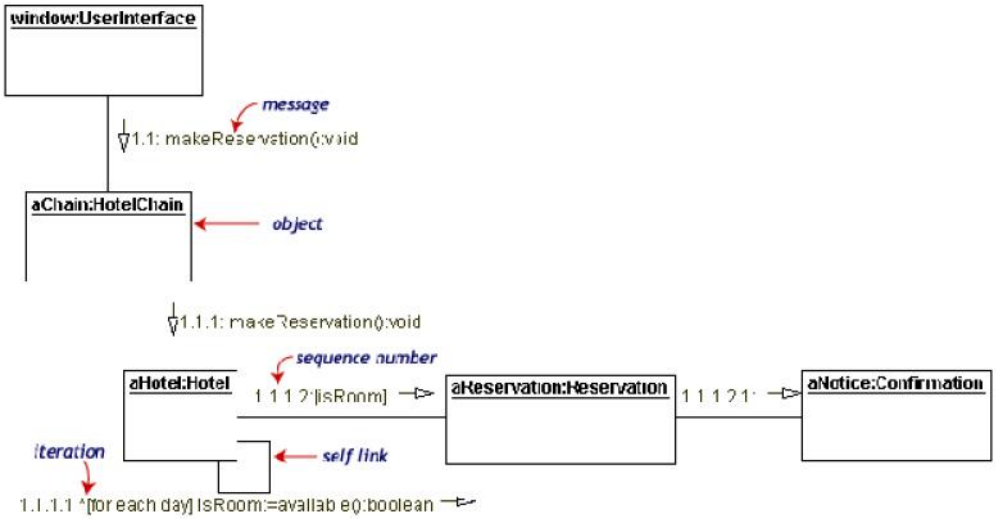
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

3.2.3.1.2.6 Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

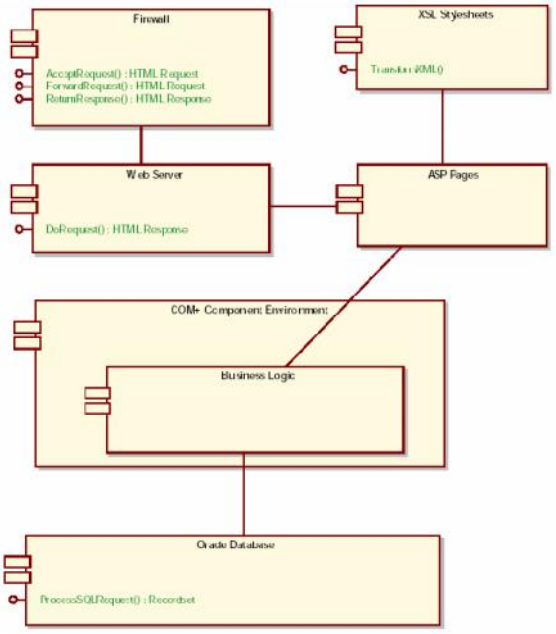


Gambar 3.4 Contoh Colaboration Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.3.1.2.7 Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

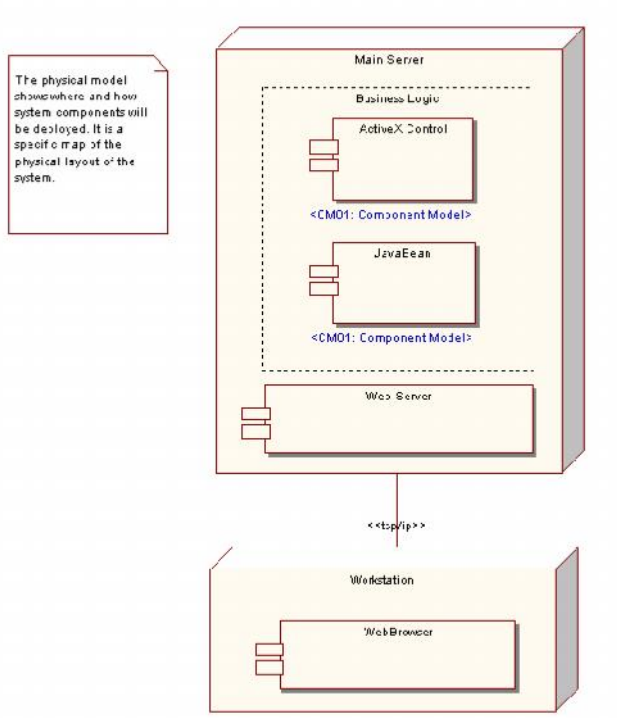


Gambar 3.5 Contoh Componen Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.3.1.2.8 Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

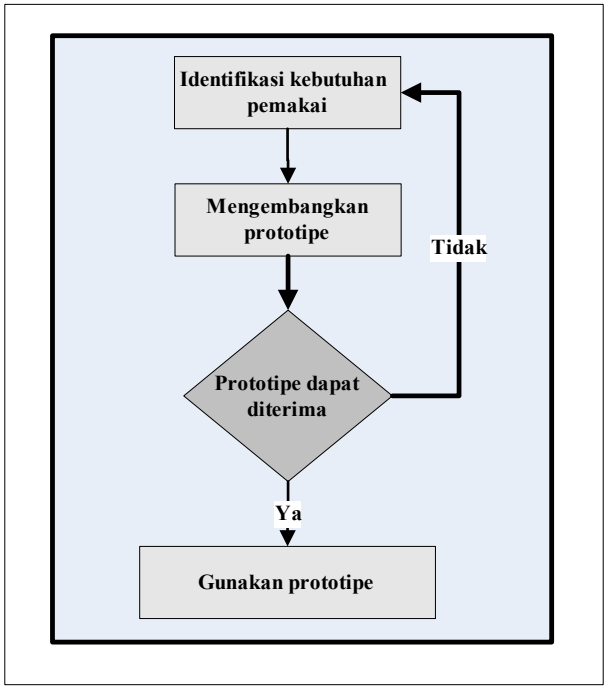


Gambar 3.6 Contoh Deployment Diagram

(Booch Jcobson Tumbaugh, Martin Fowler , 2005, *UML Distilled* Edisi 2, Andi)

3.2.3.2 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan adalah dengan menggunakan metode Model Prototype yang merupakan mekanisme untuk mengidentifikasi kebutuhan perangkat lunak.



Gambar 3.7 Metode Pengembangan Prototype

(Sumber : Raymond McLeod, Jr. and Goorge P. Schell **.** 2007. *Management Information Systems* .New Jersey.)

a. Keuntungan prototype

1. Kesalah pahaman antara sistem developer dan sistem user dapat dididentifikasi dan dibetulkan.
2. Prototype yang sedang bekerja mungkin sangat berguna dalam suatu pembuktian manajemen dimana proyek adalah fesibel sehingga menjamin kelangsungan dukungan

b. Kelemahan prototype

1. Prototype hanya akan berhasil jika pemakain bersungguh-sungguh dalam menyediakan waktu dan pikiran untuk mengerjakan prototype.
2. Kemungkinan dokumentasi terabaikan karena pengembangan lebih berkonsentrasi pada pengujian dan pembuatan prototype.
3. Mengingat target waktu yang pendek, ada kemungkinan sistem yang dibuat tidak lengkap dan bahkan kurang teruji.
4. Jika terlalu banyak proses pengulangan dalam membuat prototype, ada kemungkinan pemakai merasa jenuh dan memberikan reaksi negatif
5. Apabila tidak terkelola dengan baik, prototype menjadi tidak pernah berakhir. Hal ini disebabkan permintaan perubahan terlalu mudah untuk dipenuhi

3.2.3.3 Alat Bantu Analisis

OOP adalah suatu model pemrograman yang berorientasi pada obyek. Maksudnya adalah, bahwa setiap bagian dari apa yang anda program -visual maupun non visual- adalah sebuah obyek. Obyek, dalam konteks ini adalah "sesuatu", bisa dianalogikan dengan benda, yang padanya melekat attribut-attribut. Attribut tersebut bisa dikelompokkan ke dalam 2 jenis yaitu kelompok "property" dan kelompok "method" atau "event"(Y. Wibisono).

3.2.4 Pengujian Software

Pengujian merupakan bagian penting dalam siklus pengembangan perangkat lunak. Pengujian dilakukan untuk menjamin kualitas dan juga mengetahui kelemahan dar perangkat lunak. Tujuan dari pengujian ini adalah untuk menjamin bahwa perangkat lunak yang dibangun memiliki kualitas yang dapat diandalkan.

Dalam suatu program membutuhkan suatu teknik pengujian yang berguna dalam mengevaluasi program yang dibuat. Adapun teknik pengujian yang digunakan dalam perangkat lunak tersebut adalah *Black box.*

3.2.4.1 *Black box*

Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran yang dihasilkan dari data atau kondisi masukan yang diberikan untuk fungsi yang ada tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut. Dari keluaran yang dihasilkan, kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus dapat diiketahui kesalahan-kesalahannya

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

a. Fungsi – fungsi yang tidak benar atau hilang.

b. Kesalahan interface.

c. Kesalahan dalam struktur data atau akses database eksternal.

d. Kesalahan kinerja.

e. Inisialisasi dan kesalahan terminasi.